A Modern Turing Test: Bot Detection in MMORPGs

Adam Cornelissen [a] Franc Grootjen [b]

[a] *ICIS, Radboud University Nijmegen, the Netherlands*
[b] *NICI, Radboud University Nijmegen, the Netherlands*

Abstract

Modern online multiplayer games have become increasingly popular with gamers all around the world. This applies in particular to the kind of games that can be played with hundreds to thousands of players simultaneously, the so called 'massively multiplayer online games', often simply referred to as MMORPGs.

In these games players play as a virtual character taking on the role of a knight, priest, mage or some other heroic character to defeat enemies, to complete tasks (widely known as 'quests') or to compete in battles with other players.

While doing this, players receive items (such as gold or potions), new equipment (such as swords, shields and armor) or increased experience (how well your character is able to do a certain task) as a reward for their effort.

Not everyone though plays according to the rules of the game. A multitude of ways to cheat in games exist. In this paper we will try to find a method to automatically detect a kind of cheating where players use programs to automate their actions; the use of game bots. The presented method will be validated using a small scale experiment of twenty-five players and the same amount of bots.

1 Introduction

Massively multiplayer online role-playing games (MMORPGs) is a genre of computer role-playing games (CRPGs) in which a large number of players interact with each other in a virtual world. MMORPGs are becoming increasingly popular with gamers around the world. Millions of players worldwide (see Figure 1) pay a monthly fee to reside in an online game environment to play together.

This billion dollar industry[1] is facing the threat of cheating players, in particular those using automated tools (bots).

The use of game bots wrecks the balance of power and economy in such games; game developers adjust their game to what they think is 'reasonable gameplay' for normal players. Bots, however, play differently: they can, for example, play extremely long or handle repetitive gameplay elements over and over again. It is also known that in-game items gathered by bots are sold on auction websites such as E-Bay, thus having an impact on not only the virtual, but also the real economy.

Most game companies tried to protect their games against botting by using *traffic encryption*. Unfortunately since the encryption code is executed client side, and therefore publicly available, these protection mechanisms only last for a couple of weeks. Members of hacker communities consider it a challenge to be the first one to distribute the new modified working bots.

1.1 Bots

Normally, a player character is operated by a human being who is playing the game. However, tools exist to let your character play automatically, without human interaction. A character not being operated by a real person but by a computer program is called a 'game bot' or simply a 'bot', which is an abbreviation for 'robot'.

[1] Revenues from U.S. online gaming services will increase from $1.1 billion in 2005 to more than $3.5 billion in 2009 [3]

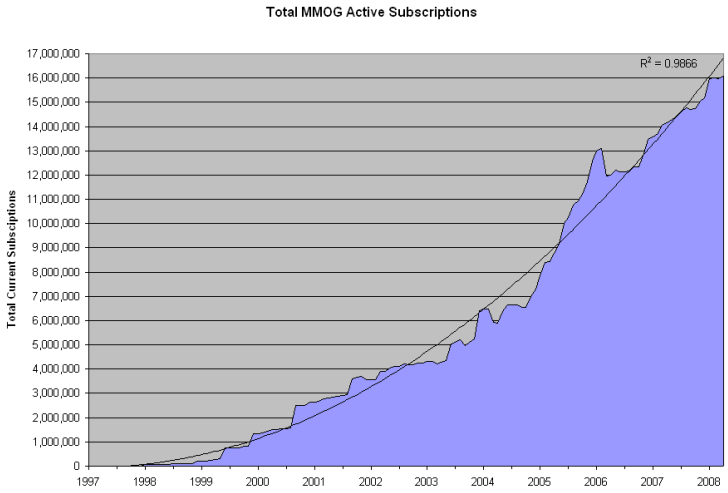

Figure 1: Total MMOG Active Subscriptions [9], these figures include *known* subscriptions to MMORPGs and do not include subscriptions to for example Ragnarok Online

1.2 Purpose of bots

Players advance in the game through completing quests, competing with other players and by defeating enemies. In the beginning every player has the weakest equipment in the game and very little experience at different actions (bad offense, bad defense, no skills such as item crafting, ...) but throughout the game players can gain better equipment, higher experience and more advanced skills.

To get higher experience players might choose to kill enemies. Certain types of enemies are often slain by using a similar strategy over and over again every time a player encounters such an enemy. While doing this, players gain experience which allows them to access newer and better game features.

People that enjoy playing the game but who simply lack the time to engage in the aforementioned repetitive gameplay might consider using bots for this repetitive and non-entertaining task.

Another reason why people use bots is to get better items and equipment. After slaying an enemy there is a chance that it will drop a certain item. Thus, if an item has a 0.01% chance to be dropped by a certain enemy, the item will drop approximately once per every 10.000 enemies slain. By using a bot, getting such an item is merely a matter of letting the bot run a long time instead of actually playing for hours.

Lack of time and/or lust to play the repetitive elements in the game and the will to easily get better ingame items and experience are the primary reasons for the use of bots.

1.3 Negative consequences of bots

At first glance it might seem that there is no reason for disallowing the usage of bots to handle the repetitive gameplay or to get items. However, there are some downfalls in allowing bots.

The use of bots can disrupts the balance of power in the game. When killing monsters over and over again (much more than one would normally do) the chances of getting a powerful item which rarely drops increases significantly.

Also, because bots can handle things repetitively and often very quickly, they can outplay human-controlled characters, giving them an unfair advantage over legitimate players.

For almost the same reason as disrupting the balance of power in the game, the use of bots can also break the economy of the game. In almost any game enemies can drop some kind of currency (cash, gold, ...). Letting a bot kill a lot of enemies gives the bot owner a lot of this currency, making him (a lot) wealthier than the average player. The process of letting a bot kill enemies to gain more ingame currency is very common and known by gamers as 'gold farming' or 'farming'.

Another reason to disallow bots is that they have a negative influence on the gaming experience of normal players. The use of bots is viewed as 'unfair' behaviour by the majority of the gaming community. Furthermore, since the game is multi user, players will encounter bots ingame. Obviously most bots are

antisocial, using shared game resources. These practices are known as 'kill stealing' and 'loot stealing'.

The most shocking fact about botting might be this: it is known that sometimes ingame items and currency gathered by bots are sold for real money on auction websites. This shows that bots do not only have an influence on the virtual economy, but also on the real economy.

For these (and other) reasons, companies running online games often disallow the use of bots in their Terms of Service (ToS) or End-User License Agreement (EULA).

1.4 Related work

Cheat prevention is regarded as a crucial challenge in the design of online games [1, 5, 6, 11]. Because game cheats often exploit loopholes in game rules or implementation bugs the main research focus is directed to correctness proofs and runtime verification of transaction atomicity [6]. Unfortunately, since bots do obey the rules of the game, these approaches do not apply to bot detection.

In a more recent study [4] Chen et al. describe an approach to identify MMORPG bots by analysing their traffic patterns. They propose strategies to distinguish bots from human players based on their traffic characteristics. The strategies are Command Timing, Traffic Burstiness, Reaction to Network Conditions. Unlike the approach presented in this paper, they do not use the *contents* of the traffic.

Yampolskiy and Govindaraju [10] as well as Golle and Ducheneaut [7] show how an embedded non-interactive test can be used to prevent bots from participating in online games. Although they report success in poker/card games, their embedded challenge response system appears to be difficult to integrate in MMORPGs without becoming annoying.

2 The Experiment

In order to create a detection method for bots, we decided to focus on 'Ragnarok Online' [8], a game known to be attacked by bots.

2.1 Ragnarok Online

Ragnarok Online, one of the most popular MMORPGs worldwide, was created by the Gravity Corporation in 2002. According to MMOGCHART [9]:

> *Ragnarok Online is supposedly the second biggest MMOG in South Korea, with well over 2 million subscribers. ... Recently they claimed 17 million worldwide with over 700,000 in North America, ...*

One reason for its popularity in Asia is the well-rendered anime style. Figure 2 shows a couple of players ingame. The game's design encourages players to get involved with other characters and form parties and guilds, which may be one of the main contributions to its popularity.

2.2 OpenKore

Unfortunately, due to its popularity there are several bot implementations for Ragnarok Online. The biggest well-known bot implementation is OpenKore, a free cross platform open source project [2]. According to the information on their own website the market share of OpenKore is about 95%. The software is highly configurable and (important for most users) runs out of the box. The availability and the easiness of use of OpenKore form the biggest threat to the game. OpenKore is the bot implementation that is used in our research. We will use the software as it is provided. This means that no changes will be made that will change the behaviour of the bot. It should be noted that the vast majority of people who use bots use it like this. The bot will automatically walk around, use teleporters, attack monsters, pick up items, sit down to recharge health, teleport back to town when very close to death and perform several other actions.

2.3 Collecting Data

Ragnarok Online consists of a client (running on the gamer's pc) which communicates with a server. The communication consists of a stream of packets which tells the server what actions the player performs. Using a network tool we were able to capture these packets, both for normal (human) players using the official

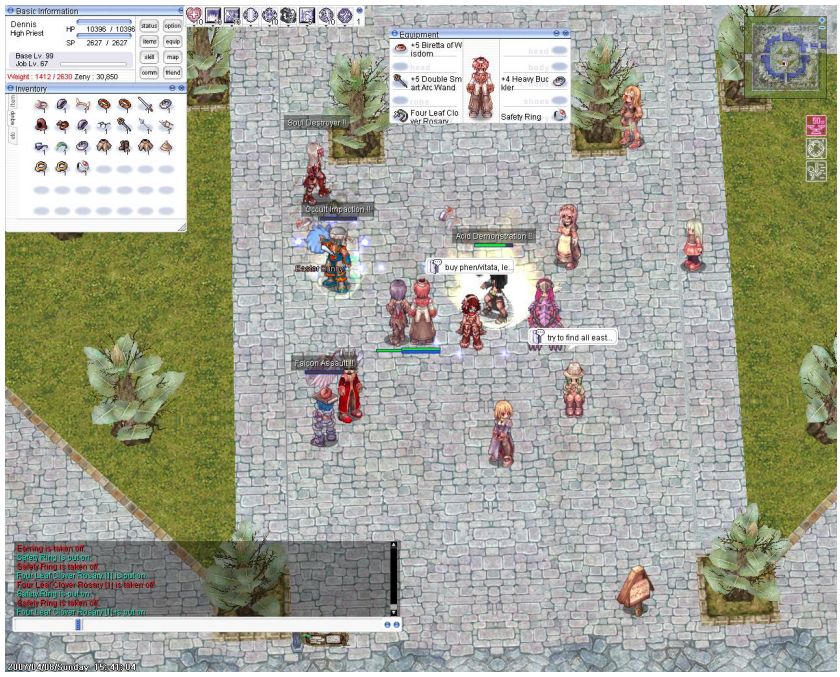

Figure 2: Screenshot from the game Ragnarok Online

client, as well as for bots driven by OpenKore. Within these packetstreams we were able to distinguish different kind of packets. Obviously we are in particular interested in packets which could help us in identifying bots and humans. Table 1 shows the packets we selected for our experiment. Along with the packet

MakeConnection	Starts a new session
MoveToXY	The player moves to a location
ChDir	Change viewing direction
TakeItem	The players picks up an item
Attack	The player attacks (a monster)

Table 1: Selected packets for investigation

information we stored a time stamp for each packet. Using these timestamps we calculated the following derived values for each session:

- *Number of packets per second.*
  All packets are counted (not only the selected ones), and divided by the duration of the session in seconds.

- *Average time between 'Move' packets.*
  The average time between two moves. In Ragnarok Online a player moves by selecting a target with the left mouse button.

- *Average distance between coordinates of 'Move' packet.*
  The Euclidean distances between the target of two subsequent moves (in game coordinates).

- *Average time between 'Take Item' packets.*
  The average time between two pickups. Most monsters drop items when killed.

- *Average time between 'Change Direction' and 'Take Item' packet.*
  If the character is not facing an object that he/she wants to pick up, it will change direction first. This feature measures the average time between the direction change, and the actual take item packet.

- *Average time between a kill and the first following 'Take Item' packet.*
  The average time it takes for the player to pick up an item after a monster is killed.

- *Number of attacks per second.*
  The number of attacks divided by the duration of the session in seconds.

These numerical values form a feature vector which (hopefully) represents the behaviour of a player.

Of course these features were selected carefully. Although all features showed tendencies which differ for bots and human players, none of them alone could determine if the session was human or bot driven. See for example Figure 3 which shows how in general bots have higher packet rates then human players, but the feature alone is not enough to decide who we are dealing with.

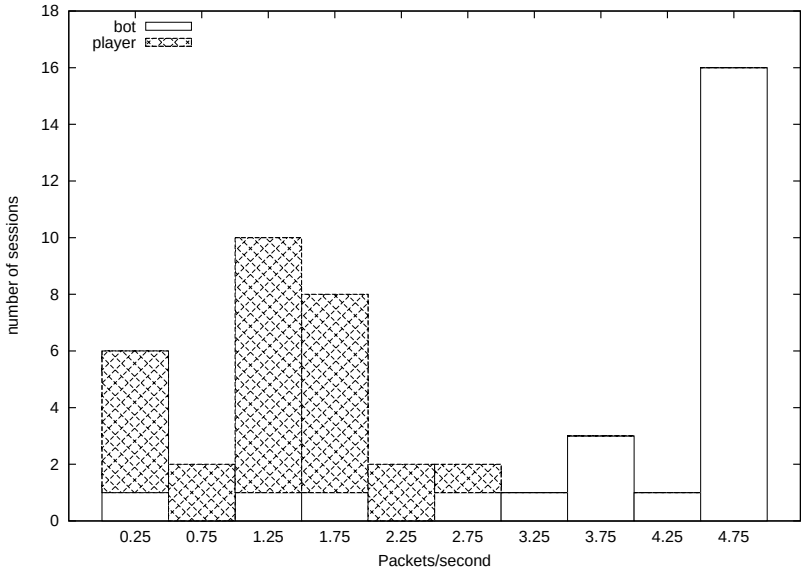

Figure 3: Histogram for number of packets/second feature

2.4 Classification

In order to classify our feature vectors we used a standard BackProp Neural Network with 7 input neurons (one for each feature), 7 hidden neurons and a single output neuron. Output activation '0' represents a player, while '1' designates a bot. We used the standard sigmoid threshold function, and learning factor $\eta = \frac{1}{2}$. We trained the network until the RMS error was below 0.0001.

To help out with the experiment we asked 25 players (of different jobs/levels) to play a session of at least 15 minutes. There were no restrictions on their behaviour whatsoever. Subsequently we configured 25 different OpenKore bots and observed them on different locations, fighting different monsters.

3 Results

Since the dataset is relatively small, we decided to create a trainingset of 49 sessions, and a testset of 1 single session. This procedure is repeated 50 times (leaving out every player/bot once). The results are presented in Table 2. The results show extremely good classification results, except for Player 13, Player 17, Bot 19 and Bot 22. Some further investigation learned that both player 13 and 17 used a special levelling technique (called mobbing) in which large groups of monsters are collected and killed together. It is possible that the character's movement during the collection phase has a mechanical touch and is misinterpreted by

test	activation	test	activation
Player 1	0.0004	Bot 1	0.9999
Player 2	0.0006	Bot 2	0.9997
Player 3	0.0005	Bot 3	0.9999
Player 4	0.0010	Bot 4	0.9999
Player 5	0.0003	Bot 5	0.9999
Player 6	0.0053	Bot 6	0.9999
Player 7	0.0003	Bot 7	0.9971
Player 8	0.0003	Bot 8	1.0000
Player 9	0.0002	Bot 9	0.9994
Player 10	0.0013	Bot 10	0.9999
Player 11	0.0004	Bot 11	0.9998
Player 12	0.0007	Bot 12	0.9999
Player 13	0.9975	Bot 13	1.0000
Player 14	0.0004	Bot 14	1.0000
Player 15	0.0007	Bot 15	0.9997
Player 16	0.0010	Bot 16	0.9999
Player 17	0.7023	Bot 17	0.9998
Player 18	0.0002	Bot 18	0.9999
Player 19	0.0007	Bot 19	0.0098
Player 20	0.0008	Bot 20	0.9999
Player 21	0.0008	Bot 21	0.9999
Player 22	0.0009	Bot 22	0.6731
Player 23	0.0001	Bot 23	0.9999
Player 24	0.0006	Bot 24	0.9999
Player 25	0.0007	Bot 25	0.9841

Table 2: Output activations for players/bots

the network. Both Bot 19 as 22 appeared to have rested long periods during their sessions. Since there was one player who spent her complete session sitting, it is possible that in those cases not enough feature information was present to enable proper determination.

All scores combined yield an average score of 94%. Thresholding at 0.5 would mean 4 errors in 50, so 92%.

4 Conclusions

Although the method presented in this paper is only validated in a small scale experiment, the results look promising. Knowing that (for now) only simple features were used and the session times were short, there appears to be enough room for improvements. A valuable feature might be the average angle between moves: bots often appear to make strange course corrections, while human players tend to move smoother, according to a plan.

Obviously, when bot developers find out what aspects of their bot makes it detectable by our method, they can adjust their bot to make it undetectable again. We do not consider this a weakness of our particular detection method. It is a mere fact that when a bot developer knows what it's bot is being tested on, they can make it undetectable again. Obviously, if a bot can mimic human actions perfectly, there is no way to distinguish a bot from a player by looking at it's behaviour.

Future research might consider the asymmetry in the decision making: having false positives (recognizing players as bots) is worse then having false negatives (bots are recognized as human players). For example, if the network's outcome is connected to an automatic jailing system[2] people might be rightfully upset if they are jailed without doing anything wrong.

[2]Ragnarok Online has a jail, in which players are placed if they violate the rules.

5 Acknowledgements

We like to thank Alchemist, Aligner, BamBam, buf priest, Cartimandua, dark acolyte, Datsylel, Ewan, Frea, Henkie, icarus, ilene, Kugutsu, maat van Ruben, Ophelia, Othello, Pebbles, Raio, Rhytor, Ruin, Rydia, Scarva, sniperboy, sniperboy's priest, and Xena for helping out with the experiments.

References

[1] N.E. Baughman and B.N. Levine. Cheat-proof playout for centralized and distributed online games. In *Proceedings of IEEE INFOCOM 2001*, pages 104–113, 2001.

[2] The OpenKore Project (Ragnarok Online Bot). `http://www.openkore.org`, last accessed August 2008.

[3] Y. Cai and P. Shackelford. *Networked Gaming: Driving the Future*. Parks Associates, Dalles, Texas USA, 2005.

[4] Kuan-Ta Chen, Jhih-Wei Jiang, Polly Huang, Hao-Hua Chu, Chin-Laung Lei, and Wen-Chin Chen. Identifying MMORPG bots: a traffic analysis approach. In *ACE '06: Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology*, pages 4–12, New York, NY, USA, 2006. ACM.

[5] E. Cronin, B. Filstrup, and S. Jamin. Cheat-proofing dead reckoning multiplayer games. In *Proceedings of the* 2^{nd} *International Conference on Application and Development of Computer Games*, 2003.

[6] M. DeLap, B. Knutsson, H. Lu, O. Sokolsky, U. Sammapun, I. Lee, and C. Tsarouchis. Is runtime verification applicable to cheat detection? In *Proceedings of ACM SIGCOMM 2004 Workshops on NetGames '04*, pages 134–138, 2004.

[7] Philippe Golle and Nicolas Ducheneaut. Preventing bots from playing online games. *Comput. Entertain.*, 3(3):3–3, 2005.

[8] Ltd. Gravity Co. Ragnarok online. `http://www.ragnarokonline.com`, last accessed August 2008.

[9] B.S. Woodcock. An analysis of MMOG subscription growth. `http://www.mmogchart.com`, last accessed August 2008.

[10] Roman V. Yampolskiy and Venu Govindaraju. Embedded noninteractive continuous bot detection. *Computers in Entertainment*, 5(4):1–11, 2007.

[11] Jeff Yan and Brian Randell. A systematic classification of cheating in online games. In *NetGames '05: Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, pages 1–9, New York, NY, USA, 2005. ACM.